Docker: Beyond the Basics

CI/CD (Day One)

Release 2021-05-29-1039

Instructor

Sean P. Kane

@spkane

@superorbital_io

SUPERORBITAL

Follow Along Guide

Textual Slides

Prerequisites (1 of 2)

- A recent computer and OS
  - Recent/Stable Linux, macOS, or Windows 10
  - root/admin rights
  - Sufficient resources to run one 2 CPU virtual machine (VM)
  - CPU Virtualization extensions MUST be enabled in your BIOS/EFI
  - Reliable and fast internet connectivity
- Docker Community/Desktop Edition & Docker Compose

Prerequisites (2 of 2)

- A graphical web browser
- A text editor
- A software package manager
- Git client
- General comfort with the command line will be helpful.
- [optional] tar, wget, curl, jq, SSH client

A Note for Powershell Users

Terminal commands reflect the Unix bash shell. PowerShell users will need to adjust the commands.

- Unix Variables
  - `export MY_VAR=test`
  - `echo ${MY_VAR}`
- Windows 10 Variables (powershell)
  - `$env:my_var = "test"`
  - `Get-ChildItem Env:my_var`

Translation Key

'/' - Unix Shell Line Continuation
'`' - Powershell Line Continuation (sort of)

${MY_VAR} - Is generally a place holder in the slides.

Linux Container Mode

- On the Windows platform make sure that you are running in Linux Container mode.

A Note About Proxies

Proxies can interfere with some Docker activities if they are not configured correctly.

If required, you can configure a proxy in Docker Desktop via the preferences.

- Docker
- Docker-Compose

Instructor Environment

- **Operating System**: macOS (v11.2.X+)
- **Terminal**: iTerm2 (Build 3.X.X+) - https://www.iterm2.com/
- **Shell Prompt Theme**: Starship - https://starship.rs/
- **Shell Prompt Font**: Fira Code - https://github.com/tonsky/FiraCode
- **Text Editor**: Visual Studio Code (v1.X.X+) - https://code.visualstudio.com/

Docker in Translation

- **Docker client**
    - The docker command used to control most of the Docker workflow and talk to remote Docker servers.
- **Docker server**
    - The dockerd command used to launch the Docker daemon. This turns a Linux system into a Docker server that can have containers deployed, launched, and torn down via a remote client.

Docker in Translation

- **Virtual Machine**
  - In general, the docker server can be only directly run on Linux. Because of this, it is common to utilize a Linux virtual machine to run Docker on other development platforms. Docker Community/Desktop Edition makes this very easy.

Docker in Translation

- **Docker images**
  - Docker images consist of one or more filesystem layers and some important metadata that represent all the files required to run a Dockerized application. A single Docker image can be copied to numerous hosts. A container will typically have both a name and a tag. The tag is generally used to identify a particular release of an image.

Docker in Translation

- **Linux Containers**
    - A Linux Container is a single instantiation of a Docker or OCI-standard image. A specific container can only exist once; however, you can easily create multiple containers from the same image.
    - OCI - Open Container Initiative

The Slow Rise Of Linux Containers

- **chroot system call** (*1979*)
  - Version 7 Unix
- **jail** (*2000*)
  - FreeBSD 4.0
- **Solaris Zones** (*2004*)
  - Solaris 10
- **Linux Containers - LXC** (*2008*)
  - version 2.6.24 of the Linux kernel

Docker Engine isn’t a…

- virtualization platform (VMware, KVM, etc.)
- cloud platform (AWS, Azure, etc.)
- configuration management tool (Chef, Puppet, etc.)
- deployment framework (Capistrano, etc.)
- development environment (Vagrant, etc.)
- workload management tool (Mesos, Kubernetes, etc.)

Linux Namespaces

- Mount (filesystem resources)
- UTS (host & domain name)
- IPC (shared memory, semaphores)
- PID (process tree)
- Network (network layer)
- User (user and group IDs)

Control Groups (cgroups)

- Resource limiting
- Prioritization
- Accounting
- Control

Testing the Docker Setup

```
$ docker image ls
$ docker container run -d --rm --name quantum \
    --publish mode=ingress,published=18080,target=8080 \
    spkane/quantum-game:latest
$ docker container ls
```

- In a web browser, navigate to port 18080 on your Docker server.

```
$ docker container stop quantum
$ docker container ls
$ docker container ls -a
```

Exploring the Docker VM

- Based on Alpine Linux (apk)

```
$ docker container run -it  --privileged --pid=host debian \
    nsenter -t 1 -m -u -n -i sh
# cat /etc/os-release
# exit
```

- http://man7.org/linux/man-pages/man1/nsenter.1.html

```
$ docker container run -ti --rm spkane/quantum-game:latest \
    cat /etc/os-release
```

Setting the Stage

```
$ cd ${HOME}
$ mkdir class-docker-cicd
$ cd class-docker-cicd
$ mkdir code
$ git clone https://github.com/spkane/docker201.git layout \
    --config core.autocrlf=input
$ cd layout
$ ls
```

Automating Workflow

- Datastore
  - Postgres
- Collaborative Source Code Repository
  - Gogs
- Docker Image Repository
  - Docker Distribution
- Build, Test, and Deploy
  - Jenkins

Iterative Workflow

- Core Technology - Docker

User develops code locally (Docker)
User commits code (Gogs backed by Postgres)
Pipeline builds & tests code (Jenkins & Docker Distribution)
Pipeline deploys code to production.
and then iterate...

Composing a Docker Service

For unix: `$ alias dc='docker-compose'`
For Windows Powershell: `PS C:\> New-Alias dc docker-compose.exe`

- Open & explore docker-compose.yaml in your text editor
- Full Documentation:
  - https://docs.docker.com/compose/compose-file/

Creating a Datastore

```
$ cd compose/review/1st
$ vi docker-compose.yaml
```

- Note: DB user & password

Creating a Source Repo

```
$ cd ../2nd
$ vi docker-compose.yaml
```

Docker Distribution

```
$ cd ../3rd
$ vi docker-compose.yaml
```

Important Note For Windows Users

- In the next section you might see:
  - a Windows Security Alert for `vpnkit.exe`, be sure and select `Allow access`.
  - Multiple `Docker for Windows - Share drive` alerts. Be sure and select `Share it` for each prompt.
- If you have problems with file mounts you may need to set: `$Env:COMPOSE_CONVERT_WINDOWS_PATHS=1`

Manage Secrets

```
$ cd ../../final
$ echo 'MY_PG_PASS=myuser-pw!' > ./.env
```

- On Windows try:
  - `Add-Content ./.env "MY_PG_PASS=myuser-pw!"`

Jenkins

```
$ vi docker-compose.yaml
$ docker-compose config
$ docker-compose up -d
$ docker-compose ps
$ docker-compose logs -f
2017/07/01 20:06:31 [ INFO] Listen: http://0.0.0.0:3000
LOG: database system is ready to accept connections
msg="debug server listening localhost:5001"
Please use the following password to proceed to installation
```

Important

Note: Don't run `docker-compose down` until class is over.

Configure Gogs

- Navigate web browser to:
  - http://127.0.0.1:10080/install
- **Database Type**: `Postgresql`
- **Host**: `postgres:5432`
- **User**: `postgres`
- **Password**: `myuser-pw!`
- **SSH Port**: `10022`
- **Application URL**: `http://127.0.0.1:10080/`

Create Gogs User

- Click: **Admin Account Settings**

Username: `myuser`
Password: `myuser-pw!`
Confirm Password: `myuser-pw!`
Email Address: `myuser@example.com`
Click: `Install Gogs`

Create GIT Repo

Click: `+`
Click: `+ New Repository`
Repository Name: `outyet`
Click: `Create Repository`

Git Credentials

- In the next section **Windows users** will likely see a GUI based password prompt from git.
- **Unix users** will likely just see a text based prompt.
- Be sure to provide your gogs username and password for the prompt.

Explore the Code

```
$ cd ~/class-docker-cicd/code/outyet
```

- Explore with your favorite code editor
    - Dockerfile
    - main.go
    - main_test.go

```
docker-compose up -d
```

Examine Application

- Navigate web browser to:
    - http://127.0.0.1:10088/

```
$ docker-compose down
```

First Code Commit

```
$ cd ../../..
$ cp -a outyet ../code/
$ cd ../code/outyet/
$ git init
$ git config core.autocrlf input
$ git add .
$ git commit -m "first commit"
$ git remote add origin http://127.0.0.1:10080/myuser/outyet.git
$ git push -u origin master
```

- username: `myuser`
- password: `myuser-pw!`

Docker Distribution Login

```
$ docker login 127.0.0.1:5000
```

- username: `myuser`
- password: `myuser-pw!`

NOTE: The example registry TLS certificate includes a SAN for `private-registry.localdomain`. You can add an entry in `/etc/hosts` or `C:\Windows\System32\Drivers\etc\hosts` to point this domain name at a remote IP address if needed.

Test Docker Distribution

```
$ docker image pull spkane/quantum-game:latest
$ docker image ls spkane/quantum-game:latest
$ docker image tag ${IMAGE_ID} 127.0.0.1:5000/myuser/quantum-game:latest
$ docker image push 127.0.0.1:5000/myuser/quantum-game:latest
```

Configure Jenkins

```
cat ../../layout/jenkins/data/secrets/initialAdminPassword
```

- Navigate web browser to:
  - http://127.0.0.1:10081/
- Paste Administrator Password

  Click: `Continue`

  Click: `Select plugins to install`

  Click: `None`

  Click: `Install`

Configuring Jenkins

- Create Admin User
  Username: `myuser`
  Password: `myuser-pw!`
  Confirm password: `myuser-pw!`
  Full Name: `My User`
  E-Mail Address: `myuser@example.com`
  Click: `Save and Continue`

Configuring Jenkins

- Final Details
  Jenkins URL: `http://127.0.0.1:10081/`
  Click: `Save and Finish`
  Click: `Start Using Jenkins`

Shutdown Services

```
$ cd ~/class-docker-cicd/layout/compose/final
$ docker-compose stop
```

Components Assembled

- Postgres Database
  - https://www.postgresql.org/
- Gogs - Source Code Manager
  - https://gogs.io/
- Docker Distribution
  - https://github.com/docker/distribution
- Jenkins CI
  - https://jenkins.io/

What We Have Learned

- Docker Compose
- Building / Running
- Ports, Volumes, and Networks
- Launched and configured:
    - Postgres / Gogs
    - Docker Distribution
    - Jenkins

Additional Learning Resources

https://learning.oreilly.com/

Any Questions?

[Sean P. Kane]

SUPERORBITAL

Providing stellar Kubernetes engineering and workshops.

https://superorbital.io/contact/